AMENDMENTS TO THE CLAIMS

1. (currently amended): A method comprising:

determining an amount of time to communicate a message and receive a response to the message by a first process respectively to and from a second process;

computing a buffer delay time from the amount of time;

storing data from the first process in a buffer; and

when the buffer delay time is reached, ~~sending~~ making the data in the buffer available to the second process by passing control of the buffer to the second process without communicating the data by the first process.

2. (currently amended): The method as described in claim 1, further comprising when the buffer is ~~approximately~~ full, sending the buffer to the second process, otherwise, when the buffer delay time is reached, making the data in the buffer available to the second process by passing control of the buffer to the second process without communicating the data by the first process.

3. (original): The method as described in claim 1, wherein determining includes:

forming a communication to send the message from the first process to the second process;

receiving the response to the message by the first process from the second process; and

monitoring a timer in relation to the communicating and the receiving to determine the amount of time.

4. (currently aemnded): The method as described in claim 1, wherein the buffer delay time is computed to be ~~approximately~~ double the amount of time.

5. (original): The method as described in claim 1, further comprising allocating the buffer using a buffer size table, wherein:

the buffer size table has a plurality of entries;

the buffer is allocated based on the plurality of entries; and

each said entry describes an amount of another buffer used to store data from the first process.

6. (original): The method as described in claim 1, wherein the first and second processes are respective running programs that communicate, one to another.

7. (original): The method as described in claim 1, wherein the first and second processes are respective running programs having one or more

corresponding sets of data that are associated with respective first and second applications.

8. (original): The method as described in claim 1, wherein:

the first and second processes are running programs that are executed on respective first and second clients; and

the first client is communicatively coupled to the second client over a network.

9. (currently amended): The method as described in claim 1, wherein the sending includes passing control over the data in the buffer from the first process to the second process by the first process by using a send operation.

10. (currently amended): The method as described in claim 1, wherein the ~~sending~~ making includes communicating the stored data within the buffer to the second process.

11. (original): One or more computer readable-media comprising computer executable instructions that, when executed on a computer, direct the computer to perform the method of claim 1.

12. (currently amended): A method comprising:

sending a message from a first process addressed to a second process;

receiving, at the first process, a response to the message sent from the second process to the first process;

computing a buffer delay time as a factor of the time between the communicating and the receiving; and

making data from the first process that is stored in the buffer available to the second process when the buffer delay time is reached by passing control of the buffer from the first process to the second process without communicating the data by the first process.

13. (currently amended): The method as described in claim 12, wherein when the buffer delay time has not been reached and the buffer is ~~approximately~~ full, sending the buffer to the second process.

14. (currently amended): The method as described in claim 12, wherein the buffer delay time is computed to be ~~approximately~~ double a time taken between the communicating and the receiving.

15. (original): The method as described in claim 12, further comprising during the receiving, storing additional data from the first process in a second said buffer.

16. (original): The method as described in claim 12, further comprising allocating the buffer using a buffer size table, wherein:

the buffer size table has a plurality of entries;

the buffer is allocated based on the plurality of entries; and

each said entry describes an amount of the memory of another buffer that was used to store data from the first process.

17. (original): The method as described in claim 12, wherein the first and second processes are at least one of:

executions of respective first and second applications; and

executed on respective first and second clients, wherein the first client is communicatively coupled to the second client over a network.

18. (currently amended): The method as described in claim 12, wherein the data stored in the buffer is available to the second process by passing control of the buffer from the first process to the second process without communicating the data by the first process by executing a send operation.

19. (original): The method as described in claim 12, wherein the data stored in the buffer is available to the second process by communicating the stored data from the buffer to the second process.

20. (original): One or more computer readable-media comprising computer executable instructions that, when executed on a computer, direct the computer to perform the method of claim 12.

21. (original): A client comprising:

a processor; and

memory configured to maintain:

one or more programs that are executable on the processor to provide respective one or more processes to process data;

a buffer that is suitable to store said data;

a buffer delay time; and

an InterProcess Control (IPC) manager that is executable on the processor to:

compute the buffer delay time from an amount of time taken to receive a response to a message by one said process from another said process; and

manage the buffer such that when the buffer delay time is reached, data stored in the buffer from the one said process is made available to the other said process by passing control of the buffer from the one said process to the other said process without communicating the data by the one said process.

22. (original): The client as described in claim 21, further comprising a timer for calculating the amount of time.

23. (original): The client as described in claim 21, wherein the IPC manager is executable in native code to perform the computing and the managing.

24. (currently amended): The client as described in claim 21, wherein the IPC manager manages the buffer such that when the buffer is ~~approximately~~ full, the buffer is sent to the other said process, otherwise when the buffer delay time is reached, the data stored in the buffer from the one said process is made available to the other said process by passing control.

25. (currently amended): The client as described in claim 21, wherein the buffer delay time is computed to be ~~approximately~~ double the amount of time.

26. (currently amended): The client as described in claim 21, wherein the data stored in the buffer is made available by passing control of the buffer from the one said process to the other said process without communicating the data by the one said process by executing a send operation.

27. (original): The client as described in claim 21, further comprising a buffer size table having a plurality of entries, each said entry describing an amount of the memory used to store data previously output by the one said process, wherein the IPC manager allocates the buffer based on the plurality of entries.

28. (original): The client as described in claim 21, further comprising a buffer delay table having a plurality of entries, each said entry describing a buffer delay time that was previously computed by the IPC manager, wherein the IPC manager computes the buffer delay time from the amount of time and the previously computed buffer delay times.

29. (original): The client as described in claim 21, wherein the IPC manager is executable to cancel the processing performed by the second process in response to a communication from the first process.

30. (currently amended): A system comprising:

a first process for outputting data;

a second process for processing the data to produce a response;

a buffer for storing the data that is shared by the first process and the second process;

a buffer delay time computed from an amount of time taken to perform the outputting and to receive the response by the first process; and

an IPC manager for managing the buffer such that when the buffer delay time is reached, another said data stored in the buffer is accessible by the second process.

31. (original): The system as described in claim 30, further comprising a timer for calculating the amount of time.

32. (original): The system as described in claim 30, wherein the IPC manager is executable in native code.

33. (currently amended): The system as described in claim 30, wherein the IPC manager manages the buffer such that when the buffer is ~~approximately~~ full, the buffer is sent to the second process, otherwise when the buffer delay time is reached, another said data stored in the buffer is accessible by the second process.

34. (original): The system as described in claim 30, wherein the IPC manager supports cancellation of the processing performed by the second process in response to a communication received from the first process.

35. (currently amended): The system as described in claim 30, wherein the buffer delay time is computed to be ~~approximately~~ double the amount of time.

36. (original): The system as described in claim 30, further comprising a buffer size table having a plurality of entries, each said entry describing an

amount of the memory used to store data previously output by the first process, wherein the IPC manager allocates the buffer based on the plurality of entries.

37. (original): The system as described in claim 30, further comprising a buffer delay table having a plurality of entries, each said entry describing a buffer delay time that was previously computed by the IPC manager, wherein the IPC manager computes the buffer delay time from the amount of time and the previously computed buffer delay times.

38. (currently amended): A system comprising:

means for providing data;

means for processing the data to produce a response for receipt by the providing means;

means for storing the data;

means for computing a delay time that is double ~~from~~ an amount of time taken to perform the outputting and to receive the response by the providing means; and

means for managing the storing means such that when the delay time is reached, another said data stored in the storing means is accessible by the processing means.

39. (original): The system as described in claim 38, wherein the providing means and the processing means are executable on a single client.

40. (original): The system as described in claim 38, wherein the providing means is executable on a client and the processing means is executable on a remote client that is communicatively coupled to the client over a network.